Multinomial Naïve Bayes or RobBERT for text classification: a customer experience case study

Fenna Anna Maria Blom

Joint Degree Bsc. Data Science
June 2021

Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands

Tilburg Law School
Tilburg University
Tilburg, The Netherlands

Abstract

To gain insights in the customer experience (CX), companies often ask for feedback. This feedback can be related to different CX metrics, that can lead to a better understanding of customer's contentment. Nowadays, CX datasets containing such feedback can be of such size that it is sensible to classify the data into different categories with the use of text mining. This thesis is a case study, evaluating Multinomial Naïve Bayes (MNB) compared to RobBERT for the classification of Dutch CX data. It does not only take performance into account, but also how explainable the techniques are. The fact that RobBERT outperformed MNB, combined with the nature of the used dataset, implies that RobBERT is the best pick.

Keywords: Text mining; Multi-class classification; Customer experience; Machine learning; Deep learning; Case study

1 Introduction

The way that companies have to compete with each other has changed over the last decades. Not only is it relevant for customers what kind of products or services are available, but also what experience comes with it. Pine and Gilmore even speak of an 'experience economy' (1998). There are multiple touch points through various channels within one customer journey. These touch points can be very important for companies in order to improve the customer experience, which is why companies have to focus on it more and more (Lemon & Verhoef, 2016).

This means that companies should not only look at what exactly their good or service is, but should also consider gathering and analyzing data, take the interaction with their customers into account, and focus on the experience itself, next to the product and the creative design (Edelman & Singer, 2015). Thus, managing the customer experience and journey has become more complex.

But what exactly is the customer experience (CX)? Lemon and Verhoef have defined it as "a multidimensional construct focusing on a customer's cognitive, emotional, behavioral, sensorial, and social responses to a firm's offerings during the customer's entire purchase journey" (2016). Even though this definition is directed at purchase, CX can be broader than that, taking any interaction between a customer and a company into account. It should be analysed, which can be done by means of so-called CX-metrices, that give insight into customer contentment. One is the Net-Promoter Score (NPS), as proposed by Reichheld. This score is about the question how likely customers are to recommend the service or good to others, which is proven to be very important for growth (2003). A different measurement is the Customer Effort Score (CES), which considers how much effort the customer had to put in throughout the customer journey, or within a specific touch point in this journey. This can identify actions to be taken by the company to make the process easier and therefore can increase the customer loyalty (Clark & Bryan, 2013). A third type of CX-metric is CSAT, which measures the customer satisfaction. It possibly focuses on specifically the good or service, but can give an evaluation about any aspect of the touch point of one's customer journey (Mittal & Frennea, 2010).

With a CX-metric, an organisation can map how well they are performing with regard to the customer experience. To gain even more insights, businesses often give their customer the option to further elaborate on their score. This can illustrate what part of the touch-point or journey the customer exactly is giving their opinion on. To have an overview of these responses, it can be useful to create clusters and even assign topics to the gathered data. This can show companies what exactly impacts these scores and can guide as a tool in deciding what to focus on in order to improve them. If the set of feedback is rather small, this can be done by hand. Otherwise it might be more effective to do it automatically, based on text mining techniques. Text mining is the process of transforming unstructured text data into structured data, which simplifies analysing this data and can help gaining information from these large datasets. These techniques can be split into two types: topic modelling and topic classification.

Topic modelling is an unsupervised form of clustering. It clusters the data into different groups, which can show what kind of topics exist in the data. Topic classification on the other hand is often a supervised technique, and is useful when it is already known what topics are expected to be found during the analysis. To either type, there are both advantages and disadvantages. Topic modelling for instance can be rather noisy, especially compared to topic classification which often leads to a better accuracy. However, topic classification tends to be more costly than modelling, because it needs manual tagging of the training data (Missier et al., 2016). In this research, the focus is on topic classification rather than topic modelling.

There are three main types of topic classification: rule-based systems, machine learning systems, and hybrid systems. Rule-based systems can work completely without machine learning. They construct a set of rules from the training data, and use these to locate information and match the textual data. Based on these rules, the test data can be classified into different topics. It is possible to achieve a good performance with this type of system, but it has its limitations. First of all, it requires a lot of work to develop good rules and systems, because they are built manually. Second of all, it requires strong, domain-specific knowledge, and the rules themselves are also domain-dependent (Vijayan, Bindu, & Parameswaran, 2017; Aggarwal & Zhai, 2012). An example of a so-called rule-based algorithm is the Ripper algorithm, which uses an information gain heuristic, and simplifies each rule by using a pruning technique. To optimise a rule set, it uses post-processing (Fürnkranz, 2013).

Machine learning systems for topic classification are mostly supervised techniques. These techniques are then again divided into three types: binary, multi-class, and multi-label classification (Gitau, 2018). Multi-class classification is for assigning one label out of multiple labels to the data. Binary classification only chooses between two different labels, and multi-label assigns multiple labels. Some of the methods that are currently most used are Naïve Bayes (NB), Support Vector Machines (SVM), and Neural Networks (NN) (Deng, Li, Weng, & Zhang, 2018). What makes NN different from the rest, is that it can be both supervised and unsupervised. The three methods all have their own benefits and drawbacks.

An NB classifier is a probabilistic learning method, which uses the training data to compute the estimations of the parameters of a probability distribution. During testing, these estimations are used to calculate the posterior probability. Then, it classifies each test sample using the hypothesis that has the highest posterior possibility. An NB classifier is relatively simple to use and has a good computational efficiency, and does not require a large training set to achieve good results. It uses a conditional independence assumption, which makes it incapable of combining multiple pieces of information together. This is something that decision trees (another algorithm used for classification) actually can do, which can lead to good results. However, training decision trees is complex and therefore computationally expensive (Spasic, Burnap, Greenwood, & Arribas-Ayllon, 2012; Daniela, Hinde, & Stone, 2009).

Initially, SVM was used for numerical data only. However, in 1997, it was found that textual data is suitable for this method as well (Joachims). It is now used in a wide variety of applications, most often because of its simple implementation, and because it is easy to interpret (Aggarwal & Zhai, 2012). The algorithm splits data into different categories by using hyperplanes (Qu, Cong, Li, Sun, & Chen, 2012). The disadvantages to this method are that it takes a lot of memory, and has a large computation time if the dataset is very large (Zhang, Li, & Yang, 2005).

There have been different studies comparing SVM and NB, often investigating either multi-class or binary topic classification. Within these tasks, differences can be found regarding the performance of the algorithms. In the research of Banik and Rahman, movie reviews from Facebook text has been classified into a positive or negative class. SVM has a precision and recall of 0.86, whereas NB achieved 84% precision and 83% recall (2018). A different study from 2018 also investigated the classification of tweets into a binary class, achieving an 88% accuracy and 88% F-measure with SVM, and 85% accuracy and 84.5% F-measure with NB (Al Hamoud, Alwehaibi, Roy, and Bikdash). The researchers of the last paper investigating the same problem, found that their SVM implementation achieved an accuracy of 83.34%, and their NB implementation 75% (Kusumawati, D'arofah, & Pramana, 2019). For these three papers, SVM has outperformed NB. This is different for multi-class classification. Wongso, Luwinda, Trisnajaya, Rusli and Rudy have classified news articles into several categories using SVM and several variations of the NB classifier, where Multinomial Naïve Bayes (MNB) achieved the best results: 98.4% precision and 98.4% recall. SVM achieved on both precision and recall 97.9% (2017). In 2018, a similar research was conducted with Turkish news texts. The MNB algorithm scored best with a performance of 89.8%, whereas SVM had only 70.6% (Gürcan, 2018). Thus, for both these studies, SVM has been outperformed by MNB.

Neural networks can represent complex relationships and retrieve good results (Daniela et al., 2009). A strong disadvantage to these networks is that it often is hard to grasp how they function exactly, which is where the term 'black box' comes from (Daniela et al., 2009). In the context of customer experience, this is important to take into account, especially if the model is being sold to clients who want to know why they receive specific outputs.

Convolutional neural networks (CNN) are a class of supervised NN, able to outperform regular NN (Deng et al., 2018). A type of unsupervised NN that has come up in the last years, and even dominates the research in machine translation, is a transformer model (Lin, Nogueira, & Yates, 2020). The Bidirectional Encoder Representations from Transformers (BERT) framework is currently the best known implementation of transformer models. It has been developed by Google, and is a pre-trained algorithm that can be optimized with adding just one output layer (Das, Mandal, & Basu, 2020). Training BERT is done by leaving out a part of the input (masking), and then letting the model predict the masked word. The input gives sequential data, where in 50% of the time not the next but a random sentence is presented. The model learns by classifying these two obstacles correctly and reaching a minimal loss for both. (Delobelle, Winters, & Berendt, 2020). This means that models in this framework actually extract features from the text themselves. Lin et al. have used BERT in their task of classifying unstructured short text data. With this model, they obtained a good accuracy of 89.67% compared to a 82.2% accuracy for both decision trees and SVM (2020).

The research question of this paper is: which machine learning technique should be used to optimally classify customer experience data? To investigate this, a comparison will be made between NB and the BERT framework. NB has been chosen over SVM, because this research focuses on multi-class classification rather than binary classification. BERT is a more advanced technique, often used to improve and better understand the word representations. A downside is that it might be seen as a black-box: it is difficult to understand how it learns exactly, especially for those who are not familiar with deep learning techniques. That is why it is compared to a more traditional technique, to make an accuracy-explainability trade-off.

2 Methods

2.1 Dataset

The dataset that is used in this research contains feedback data of a transport firm in The Netherlands, and will show which one, if any, of the investigated techniques is most usable for this type of CX classification. It is provided by Underlined, a company located in 's-Hertogenbosch, The Netherlands, focusing on providing insights in CX data.

The data consists of six columns: the parcel number, row-id, a brief feedback, a more expanded feedback, tNPS topic and tNPS subtopic. The features can be found in `Score.Explanation.Expanded`, and the target values in `TnpsTopic`. It has over 60,000 rows, of which 348 are missing data. The feedback, which is in Dutch, is an explanation on the NPS that was given in the survey. This score is not included in the dataset. The topics and subtopics come from a previous project, in which the feedback was labelled with a taxonomy-based technique. This is the same technique Underlined uses in their final product. This is why it is assumed these labels are correct.

Some part of the data was excluded from the labelling. This mostly is data that is in the predefined exclude list, and is 5.02% of the data. When it was not possible to assign a topic to the data, the data was assigned to "No topic found" and "No subtopic found". This is 4.95% of the data.

2.2 Research Methods

2.2.1 Topic Tree

As mentioned in the previous subsection, the dataset has already been labelled. These labels are in Dutch, but for this paper the topic tree is translated to English. This tree can be found in Table 2.1, and the original, Dutch topic tree can be found in the Appendix. It relates to a combination of two key performance indicators (KPIs), namely the generic one and the one for transport. The KPIs are used by Underlined to work with topic trees that are domain-specific. The topic 'No topic found' is normally used when a specific review cannot be classified into any of the existing topics.

2.2.2 Data pre-processing

Because the data that is in the predefined exclude list will not be labelled, this data will be removed from the dataset. The topic 'No topic found' will also be deleted. This is done because it is undesirable the model learns that data should be assigned to this topic when it contains specific features. By removing these two topics, the noise in the dataset will be reduced. Besides this, the topics are transformed into factors. This is necessary for models in the BERT-framework

and will be used throughout the research for consistency. In the Table 2.1 can be found which factor each category corresponds to, including the size of each class after the pre-processing.

Further pre-processing of the data will be elaborated on in the following sections.

2.2.3 Multinomial Naïve Bayes

As described in the introduction, one of the techniques investigated in this paper is NB, because it has outperformed SVM in several studies that investigated multi-class text classification. There are different types of NB classifiers, and the one that is most often used for this specific task is Multinomial Naïve Bayes, which has also been used by Wongso et al., and Gürcan (2017; 2018). It is implemented in this research as well.

For the classification of text, MNB assumes that the probability that a word occurs is independent of the appearance of the other words in that document. It calculates the probability of the document belonging in a class $(P(c_k|d_j))$, with the following equation:

$$P(c_k|d_j) = P(c_k)\prod_{i=1}^{k} P(t_i|c_k) \tag{2.1}$$

.

$P(t_i|c_k)$ denotes the probability word t_i appears in a document in class c_k. $P(c_k)$ is the prior probability a document appears in class c_k. $P(c_k)$ and $P(t_i|c_k)$ are computed based on the training data (Arora, Patel, Shaik, & Hatekar, 2020).

For implementing MNB, the `MultinomialNB` and `TfidfVectorizer` functions from the Python package sci-kit learn are used (Pedregosa et al., 2011). Because the classes in the dataset are rather imbalanced, first the top features are selected with `SelectPercentile`, then `RandomOversampler` is used in combination with `SMOTE`, and `Pipeline` is used for creating a pipeline. These three functions are from the Python package imbalanced-learn (Lemaître,

Table 2.1: Topic tree with sizes

Topic	Size
Handling (5)	9871 (23.58%)
General experience (3)	5147 (12.24%)
Delivery service (0)	12032 (28.61%)
Making contact with employee (7)	136 (0.32%)
Digital possibilities (9)	127 (0.30%)
Convenience (8)	45 (0.11%)
Attitude & behavior employee (2)	7266 (17.28%)
Information (6)	5399 (12.84%)
Knowledge & capabilities employee (4)	1328 (3.158%)
No topic found	
Price & quality (10)	148 (0.35%)
Processes (1)	621 (1.48%)

Nogueira, & Aridas, 2017). If this is not executed, MNB will assign higher weights to specific features, resulting in a 'dumb model': a model that classifies almost all data into one topic, in this case the biggest topic.

The pipeline, which connects several data processing elements so that the output of the first element is the input of the next one, is created with the functions in the following order: `TfidfVectorizer`, `SelectPercentile`, `RandomOversampler`, `SMOTE`, and `MultinomialNB`. These functions will be explained shortly, and then the parameters that are fine-tuned will be elaborated on.

`TfidfVectorizer` is for feature extraction, which evaluates the relevance of individual words or sets of words in text data. For selecting the top features, `SelectPercentile` is used. It computes and selects the top percentage with a Chi Squared test. `RandomOversampler` is for creating a more balanced class, by randomly selecting samples from the minority classes and duplicating them in the dataset. `SMOTE` is an addition to this, but instead of simply duplicating data, it is a type of data augmentation. Another way of augmenting the data could be with undersampling the data instead of oversampling it. However, this would lead to losing a big part of the data, since the smallest classes are of a completely different size than the bigger ones (45 and 127, and 12032 and 7266 respectively). The last function, `MultinomialNB`, is the function that actually executes the classification.

The functions that contain multiple parameters are `TfidfVectorizer`, `SelectPercentile`, and `MultinomialNB`. The parameters in `TfidfVectorizer` kept at default settings are `encoding`, `decode_error`, `token_pattern`, `vocabulary`, `binary`, `dtype`, `use_idf`, `smooth_idf`, and `sublinear_tf`. The remaining parameters are `strip_accents`, `preprocessor`, `stop_words`, `ngram_range`, and `norm`. `Preprocessor` is a self-written function. This is necessary, because the default pre-processor is based on English text, and the data in this research is in Dutch. The pre-processing consists of several steps. First of all, it transforms the data into lower-case text. Then, it gets rid of the punctuation that is in the data, so that it will not be extra noise in the features. After this, stop words are removed. English examples of stop words are: "she", "the", "is", "are". These stop words are imported from the nltk package (2009). Finally, the pre-processor lemmatizes the words in the data. This entails splitting the data into tokens, and then bringing it back to a standard form. E.g., "apples" would be reduced to "apple", and "walked" would be reduced to "walk". As the data is Dutch, the lemmatizer `nl_core_news_sm` from the SpaCy package is used (2017). The remaining parameters are `strip_accents`, `lowercase`, `ngram_range`. `strip_accents` is set to 'ascii'. `lowercase` is set to False, because this is already done. Lastly, `tokenizer` is set to a function that simply passes the data, because after preprocessing, the data is already tokenized.

There are two hyper-parameters in `SelectPercentile`: `score_func` and `percentile`. For `score_func`, `chi2` is selected. This means that a chi squared statistic is computed between each feature and class, and it is often used in classification models. `Percentile` decides what the percentage is for the selected features, and is further fine-tuned.

In `MultinomialNB`, there are three hyper-parameters: `alpha`, `fit_prior`, and `class_prior`. The last one is kept at the default setting, which means no prior probabilities are given to the model. `fit_prior` is set to True, meaning the prior probabilities of the classes are computed. The final parameter, `alpha`, is a smoothing parameter and is optimized with the other hyper-parameters from `TfIdfVectorizer` and `SelectPercentile`.

Table 2.2: Multinomial Naïve Bayes parameter values

Parameter	Parameter values
`norm`	'l1', 'l2'
`percentile`	$10, 15, 20$
`alpha`	$0.01, 0.1, 0.3$

The optimization of the last hyper-parameters is done with a grid search. The function used is `GridSearchCV`, also from the sci-kit learn package (Pedregosa et al., 2011). It runs all the combinations of each possible parameter value five times, and then returns the optimal hyper-parameter selection. The parameter values can be found in table 2.2.

After this, a cross-validation is executed. This allows the model to run several times, each time with different training and test samples from the data. Performing a cross-validation is done to prevent over-fitting and getting to the ultimate bias-variance trade-off. Using the `cross_validate` from the sci-kit learn package, the optimal model is selected (Pedregosa et al., 2011).

Finally, the predict function is called on this optimal model to see how well it fits the test data.

2.2.4 RobBERT

The second technique that is applied comes from the BERT framework. Because the general BERT models are mainly based on English text data, a different one that analyzes Dutch data is used. This is the RobBERT model, the state of the art model for the Dutch language. Besides the language the model is based on, a difference with the original BERT model is that RobBERT is trained with only the masking task (Delobelle et al., 2020).

Implementing this technique consists of three parts: pre-processing the data, building the model, and finally, training the model. For the biggest part, this pre-processing resembles that of MNB. The biggest difference is that the features are transformed into tokens and token ids, where each id refers to a specific word (or part of a word) in the pre-trained vocabulary. This is done with the `AutoTokenizer` imported from the RobBERT architecture (Delobelle et al., 2020).

Building the model is done by combining the pre-trained RobBERT model with additional layers, where RobBERT will function as the main layer in the model. The other layers that are used and further specified are `Input`, `DropOut` and `Dense` (Chollet et al., 2015). The dropout probability, which is set to prevent overfitting, remains 0.1, as recommended in the original BERT paper (Devlin, Chang, Lee, & Toutanova, 2019). For the final classification layer, which is a `Dense` layer that connects every neuron in previous layers to every neuron in the next layer, the parameter is kept at 1.0. The activation function in this layer is a `softmax` function.

To train the model, several functions need to be set: the optimizer, the loss function and metric. For optimising this model, two different optimizers are considered: `Adam` and `SGD`, where `Adam` automatically adapts the learning rate of the network, but in `SGD` it should be done manually (Chollet et al., 2015). The original BERT paper uses `Adam`, so this will be used in this

paper as well (Devlin et al., 2019). Since the classification is multi-class, the loss function is the categorical cross-entropy loss. This indicates how distinct two probability distributions are, leading to the categorical accuracy which is the metric. These calculations are executed with `CategoricalCrossentropy` and `CategoricalAccuracy`, both coming from the Keras library as well (2015).

Within the optimizer, different hyper-parameters need to be fine-tuned. According to Devlin et al., `batch_size`, `learning_rate`, and `epochs` are useful for improving the model (2019). Batch size is related to the number of cases that is used for learning within each iteration. The learning rate is a scalar that determines the size of the step that is taken while updating the weights during training. Epochs refer to how often the model has processed the entire dataset. The parameter settings that are used are `batch_size` = 16, `epochs` = 3, and `learning_rate` = 5e-5. These settings are based on the recommendations of Devlin et al. (2019).

After training this model, it will be used to do predictions on the test data.

3 Results

3.1 Metrics

To compare the final methods with each other based on performance, equal metrics should be used. As mentioned before, the classes in the data are not equally balanced. That is why the regular accuracy metric is not sufficient - weighted average of precision, recall and F1-score are also taken into account. These metrics all rely on the confusion matrix. This matrix will be shortly explained, after which the three metrics will be elicited.

A confusion matrix is computed per class. Each data point is considered for each class: if the data point is labelled with this class, it is predicted as true. Otherwise, it is predicted as false. The values used for computation of the metrics are:

- True positive (TP): number of instances that are correctly predicted as true.
- True negative (TN): number of instances that are correctly predicted as false.
- False positive (FP): number of instances that are falsely predicted as true.
- False negative (FN): number of instances that are falsely predicted as false.

Accuracy: What proportion of the predictions is correct?

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

Precision: What proportion of the predictions labelled true is correct?

$$PVV = \frac{TP}{TP+TN} \tag{3.2}$$

Recall: What proportion of the actual true labels is predicted correctly?

$$TPR = \frac{TP}{TP+FN} \tag{3.3}$$

F1 Score: What is the mean of precision and recall?

$$F_1 = \frac{2 \cdot TPR \cdot PVV}{TPR+PVV} \tag{3.4}$$

Macro average: This can be applied to any metric, computing these for each class individually and then taking the average.

By using the macro average along with the generic metrics, results become more representative. The use of the macro average, especially of the F1-score, is recommended by Madabushi, Kochkina, and Castelle, when evaluating predictions based on imbalanced classes (2020).

3.2 Model results

3.2.1 Multinomial Naïve Bayes

First, a grid search was executed for MNB, based on a sample of the dataset (n=10,000). As explained, this was done for finding the optimal hyper-parameter setting. The results of this can be found in Table 3.1.

Table 3.1: Results grid search Multinomial Naïve Bayes

`percentile`	`alpha`	`norm`	Mean test score (standard deviation)
10	0.01	l1	0.69 (0.010)
		l2	0.69 (0.007)
10	0.1	l1	0.70 (0.009)
		l2	0.69 (0.007)
10	0.3	l1	0.70 (0.009)
		l2	0.69 (0.004)
15	0.01	l1	0.68 (0.008)
		l2	0.67 (0.009)
15	0.1	l1	0.69 (0.006)
		l2	0.69 (0.009)
15	0.3	l1	0.69 (0.009)
		l2	0.69 (0.009)
20	0.01	l1	0.68 (0.008)
		l2	0.67 (0.007)
20	0.1	l1	0.69 (0.008)
		l2	0.68 (0.010)
20	0.3	l1	0.69 (0.007)
		l2	0.69 (0.009)

As described in the Methods section, the cross-validation is executed with the optimal hyper-parameter setting, which is `percentile`: 10; `alpha`: 0.3; `norm`: l1. This leads to an average accuracy of 0.6976(0.006).

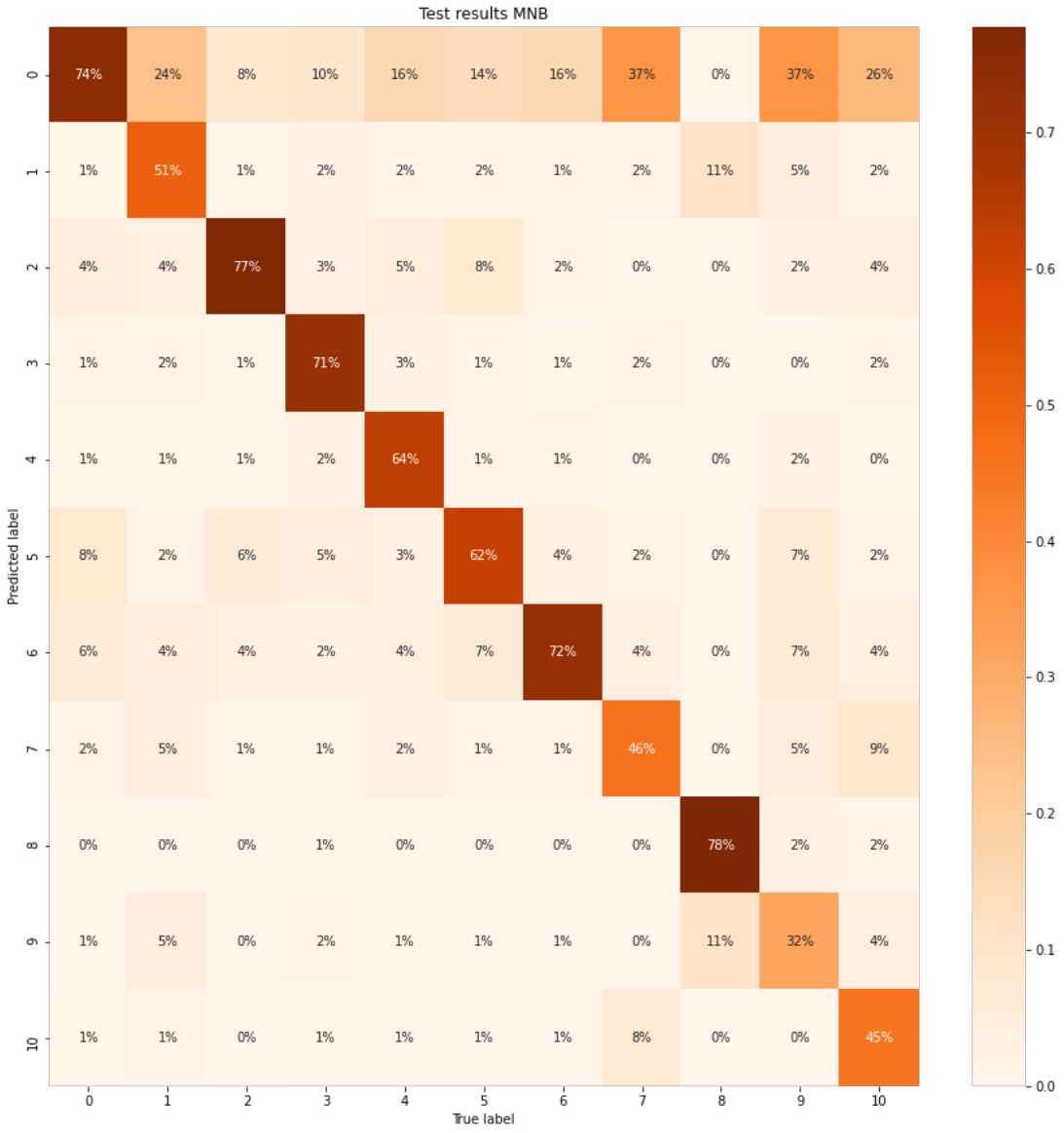
Eventually, the model is used to predict on the test data. The test evaluation can be found in Table 3.2. In Figure 3.1, a heatmap of the confusion matrix is shown.

Table 3.2: Test results Multinomial Naïve Bayes and RobBERT

	Accuracy	Precision Macro average	Recall	F1-Score
MNB	0.70	0.48	**0.61**	0.50
RobBERT	**0.84**	**0.56**	0.52	**0.51**

A part of the data has been classified incorrectly. For example, ”Geen communicatie over

Figure 3.1: Heatmap of confusion matrix Multinomial Naïve Bayes



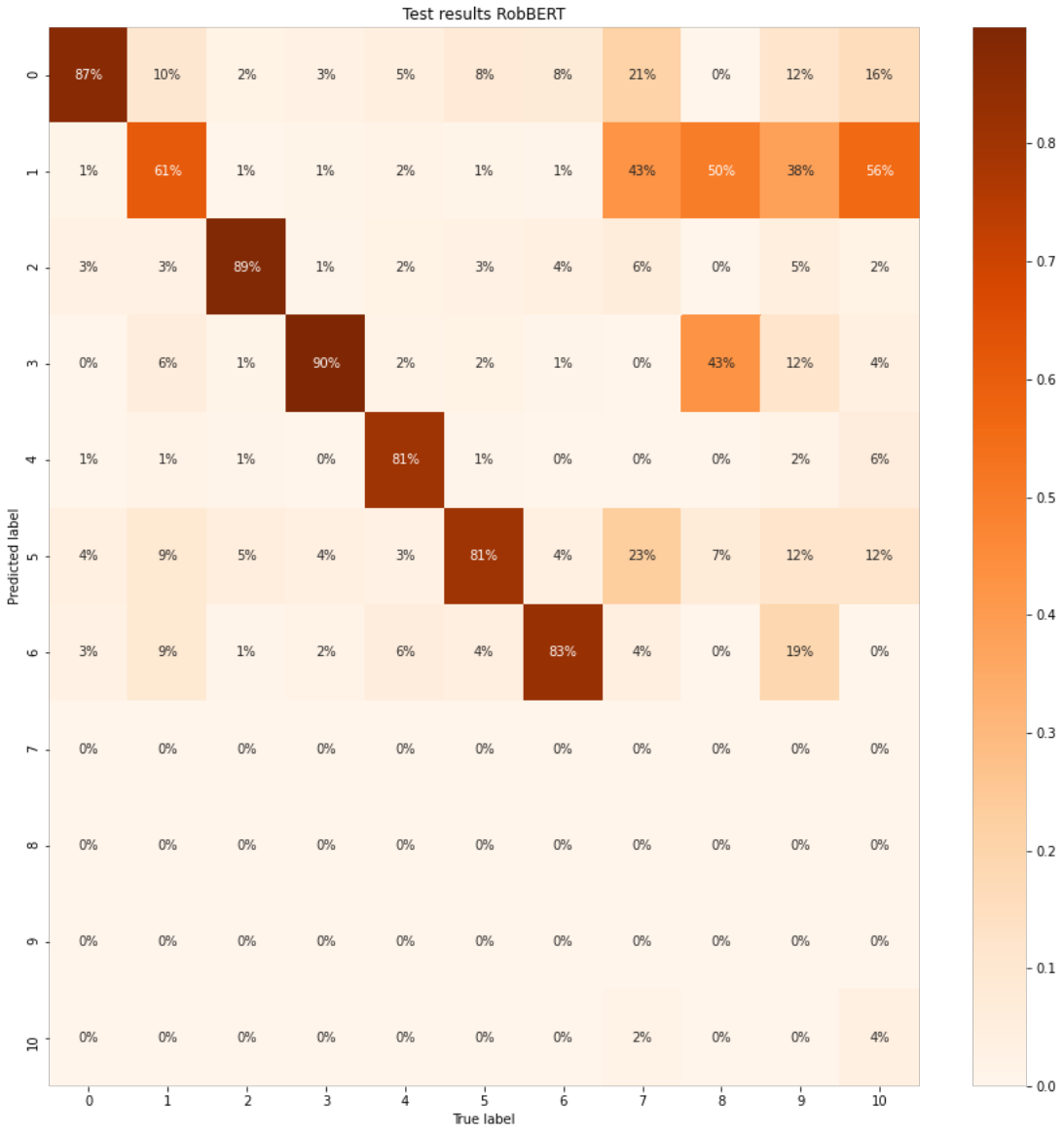
vertraging van levering 12e vertrokken uit het depot en de 19e geleverd.” which can be translated to ”No communication regarding the delay of the delivery. Left the depot at the 12th, delivered at the 19th.” was predicted to fall in Delivery service (class 0). However, it should have been Making contact with employee (class 7).

3.2.2 RobBERT

The model for RobBERT was trained on the full dataset with the parameters set as described in the Method section: `batch_size` = 16, `epochs` = 3, and `learning_rate` = 5e-5. After training, which took 16 hours and 13 minutes, the fine-tuned model was used for predicting the data. The performance evaluation of RobBERT can be found in Table 3.2, and the confusion matrix in Figure 3.2.

Again, some data was predicted wrongly. An example is ”Makkelijk verzendlabel printen, prima prijs.”, which can be translated to ”Easy to print the label, good price.” It was classified into Processes, (class 1), when it should have been Price & Quality (class 10).

Figure 3.2: Heatmap of confusion matrix RobBERT



4 Discussion

As shown in Table 3.2, MNB achieved a precision of 0.48 and a recall of 0.61. The accuracy is 0.7, and compared to the studies of Wongso et al., and Gürcan, this is a relative low performance (2017; 2018). As mentioned, the most important metric is the macro-average F1-score, which is 0.50. The results of RobBERT can be found in the same table. Accuracy is 0.84, but the other results are much lower: a macro average precision of 0.56, a recall of 0.52, and an F1-score of 0.51. On all aspects of performance evaluation except recall, RobBERT outperformed MNB. This is a valid result, since RobBERT is the state of the art model for the classification of Dutch textual data (Delobelle et al., 2020). However, it should be noted that for F1-score, the difference in performance is very small (0.01).

As the results imply, the models both have limitations. Most urgent is the bias towards the biggest classes, that is present in both implementations. This can be concluded from Figure 3.1 and Figure 3.2. With MNB, the bias is reduced compared to the default model, but it is visible that a large part of the data was predicted to fall in the largest class. In RobBERT the issue is slightly different. The model predicts none of the smaller classes correctly - they do not even occur in the predictions. An example of this can be found in section 3.2.2.

The problem with imbalanced data that occurs here, is a common one in natural language processing. Several techniques have been developed to reduce this issue, related to changing either the model or the dataset (Madabushi et al., 2020). For MNB, augmenting the data has been performed with `RandomOversampling` and `SMOTE`. This was not done for RobBERT, which indicates a big flaw in this thesis: how well RobBERT handles imbalanced data was not considered at all. If it had been taken into account and the solutions proposed by Madabushi et al. had been applied, the results would have looked differently, assumably in favor of RobBERT (2020).

5 Conclusion

In this study, two different type of machine learning techniques were investigated: Multinomial Naïve Bayes and RobBERT. The main goal was to investigate which algorithm is a better pick for the classification of CX data.

These two techniques were picked because they each represent a different category within text classification. MNB stands for more traditional machine learning, and companies have been embracing that more and more over the last years (Davenport & Bean, 2018). RobBERT on the other hand is built from the BERT framework, and this falls in the neural network category. There are some downsides to using neural networks. Where models like MNB are relatively simple to use and to explain to those less skilled in data science and artificial intelligence, this is not the case for NN (Daniela et al., 2009). The functioning of these networks is much harder to grasp. This can make it more complicated to explain to clients using such CX classification method where exactly specific outputs come from, which weighs in the accuracy-explainability trade-off. This should be taken into account even more strongly when the CX data gets more sensitive.

For CX data that does not contain information that is sensitive, like the dataset used in this thesis, accuracy can weigh more than explainability in the trade-off. Companies, especially those looking to gain insight in their CX in a more data-driven way, wish to get good results, leading them to specific points in the customer journey that have been pointed out by customers.

Despite its exploratory nature, the research gives some insights in which technique to use for classifying CX data. Since accuracy is more important than explainability in this case study, it is sensible to use RobBERT over MNB. Not only because RobBERT is currently the most advanced technique for classifying Dutch language, but also because the obtained results support this claim (Delobelle et al., 2020). Overall, RobBERT outperformed MNB. And even though the difference in F1-score is not that convincing, RobBERT shows more potential for improvement than MNB as well. This conclusion can be drawn from the fact that this research did not consider improving the class balance for RobBERT, whereas this was implemented for MNB.

How this technique can be used for the current case study is still to be determined, especially because the implementation has been far from perfect. The strongest suggestion is that RobBERT can be used for performing a quality check, this can speed up a manual quality check. Once the implementation is developed even further, one, Underlined in this case, can consider replacing the current text mining model with RobBERT.

A limitation of this study is the preliminary implementation of the techniques, however this can be passed on for future work. Delving deeper into the fine-tuning of the technique might lead to even better results. Further work can also focus on the issue with balance of the classes. Because of this, there exists a bias towards the larger classes, which might be minimised with either a method of feature selection or class balancing. For RobBERT specifically, a cost-sensitive learning technique could be realized, as suggested by Madabushi et al. (2020). Another sug-

gestion for future work is classifying the data into subtopics. This can create an even better overview of what the data is about.

References

Aggarwal, C., & Zhai, C. (2012). A survey of text classification algorithms. In C. Aggarwal & C. Zhai (Eds.), *Mining of text data* (p. 163-222). Boston, MA: Springer US. doi: 10.1007/978-1-4614-3223-4_6

Al Hamoud, A., Alwehaibi, A., Roy, K., & Bikdash, M. (2018). Classifying political tweets using naïve bayes and support vector machines. In M. Mouhoub, S. Sadaoui, O. Ait Mohamed, & M. Ali (Eds.), *Recent trends and future technology in applied intelligence* (pp. 736–744). Cham: Springer International Publishing. doi: 10.1007/978-3-319-92058-0_71

Arora, A., Patel, P., Shaik, S., & Hatekar, A. (2020). Support vector machine versus naive bayes classifier: A juxtaposition of two machine learning algorithms for sentiment analysis. *International Research Journal of Engineering and Technology*, *7*(7), 3553-3563.

Banik, N., & Rahman, M. H. H. (2018). Evaluation of naïve bayes and support vector machines on bangla textual movie reviews. In *2018 international conference on bangla speech and language processing (icbslp)* (p. 1-6). doi: 10.1109/ICBSLP.2018.8554497

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Clark, M., & Bryan, A. (2013). *Customer effort: Help or hype?* (White paper). Henley Business School.

Daniela, X., Hinde, C., & Stone, R. (2009). Naive bayes vs. decision trees vs. neural networks in the classification of training web pages. *International Journal of Computer Science Issues*, *4*(1), 16-23.

Das, S., Mandal, S., & Basu, A. (2020). Identification of cognitive learning complexity of assessment questions using multi-class text classification. *Contemporary Educational Technology*, *12*(2). doi: 10.30935/cedtech/8341

Davenport, T. H., & Bean, R. (2018). Big companies are embracing analytics, but most still don't have a data-driven culture. *Harvard Business Review*, *6*, 1–4.

Delobelle, P., Winters, T., & Berendt, B. (2020, November). RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the association for computational linguistics: Emnlp 2020* (pp. 3255–3265). Online: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/2020.findings-emnlp.292` doi: 10.18653/v1/2020.findings-emnlp.292

Deng, X., Li, Y., Weng, J., & Zhang, J. (2018). Feature selection for text classification: A review. *Multimedia Tools and Applications*, *78*, 3797-3816.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding.*

Edelman, D., & Singer, M. (2015). Competing on customer journeys. *Harvard Business Review*, *93*(11), 88-100.

Fürnkranz, J. (2013). Rule-based methods. In W. Dubitzky, O. Wolkenhauer, K. Cho, & H. Yokota (Eds.), *Encyclopedia of systems biology* (p. 1883-1888). New York, NY: Springer New York. doi: 10.1007/978-1-4419-9863-7_610

Gitau, C. (2018, Apr). *Classification in supervised machine learning: All you need to know!* Medium. Retrieved from `https://categitau.medium.com/in-one-of-my-previous-posts-i-introduced-machine-learning-and-talked-about-the-two-most-common-c1ac6e18df16`

Gürcan, F. (2018). Multi-class classification of turkish texts with machine learning algorithms. In *2018 2nd international symposium on multidisciplinary studies and innovative technologies (ismsit)* (p. 1-5). doi: 10.1109/ISMSIT.2018.8567307

Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.* (To appear)

Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the fourteenth international conference on machine learning* (p. 143–151). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kusumawati, R., D'arofah, A., & Pramana, P. (2019). Comparison performance of naive bayes classifier and support vector machine algorithm for twitter's classification of tokopedia services. *Journal of Physics: Conference Series*, *1320*, 012016. doi: 10.1088/1742-6596/1320/1/012016

Lemaître, G., Nogueira, F., & Aridas, C. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1-5. Retrieved from `http://jmlr.org/papers/v18/16-365.html`

Lemon, K., & Verhoef, P. (2016). Understanding customer experience throughout the customer journey. *Journal of Marketing*, *80*(6), 69-96. doi: 10.1509/jm.15.0420

Lin, J., Nogueira, R., & Yates, A. (2020). *Pretrained transformers for text ranking: Bert and beyond.* doi: 10.1145/3437963.3441667

Madabushi, H. T., Kochkina, E., & Castelle, M. (2020). Cost-sensitive BERT for generalisable sentence classification with imbalanced data. *CoRR*, *abs/2003.11563*. doi: 2003.11563

Missier, P., Miu, T., Pal, A., Daniilakis, M., Garcia, A., Cedrim, D., & da Silva Sousa, L. (2016). Tracking dengue epidemics using twitter content classification and topic modelling. In S. Casteleyn, P. Dolog, & C. Pautasso (Eds.), *Current trends in web engineering* (Vol. 9981, p. 80-92). Springer International Publishing.

Mittal, V., & Frennea, C. (2010). *Customer satisfaction: A strategic review and guidelines for managers.* (Working paper). Marketing Science Insititute.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pine II, B. J., & Gilmore, J. (1998). Welcome to the experience economy. *Health Forum Journal*, *76*(4), 97-105.

Qu, B., Cong, G., Li, C., Sun, A., & Chen, H. (2012). An evaluation of classification models for question topic categorization. *Journal of the American Society for Information Science and Technology*, *63*, 889-903. doi: 10.1002/asi.22611

Reichheld, F. (2003). The one number you need to grow. *Harvard Business Review*, *81*(12), 46-55.

Spasic, I., Burnap, P., Greenwood, M., & Arribas-Ayllon, M. (2012). A naïve bayes approach to classifying topics in suicide notes. *Biomedical informatics insights*, *5*, 87-97. doi: 10.4137/BII.S8945

Vijayan, V., Bindu, K., & Parameswaran, L. (2017). A comprehensive study of text classification algorithms. In *2017 international conference on advances in computing, communications and informatics (icacci)* (p. 1109-1113). doi: 10.1109/ICACCI.2017.8125990

Wongso, W., Luwinda, F. A., Trisnajaya, B. C., Rusli, O., & Rudy. (2017). News article text classification in indonesian language. *Procedia Computer Science*, *116*, 137-143. (Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017)) doi: https://doi.org/10.1016/j.procs.2017.10.039

Zhang, J., Li, Z., & Yang, J. (2005). A parallel svm training algorithm on large-scale classification problems. In *2005 international conference on machine learning and cybernetics* (Vol. 3, p. 1637-1641). doi: 10.1109/ICMLC.2005.1527207

Appendix A

Table .1: Topic tree in Dutch

Topic
Afhandeling
Algemene ervaring
Bezorgservice
Contact leggen met medewerker
Digitale mogelijkheden
Gebruiksvriendelijkheid
Houding & gedrag medewerker
Informatievoorziening
Kennis & vaardigheden medewerker
No topic found
Prijs & kwaliteit
Processen